Institute for Defense Analyses

4850 Mark Center Drive • Alexandria, Virginia 22311-1882

Linux Foundation
Core Infrastructure Initiative (CII)
Best Practices Badge

Dr. David A. Wheeler

2016-09-14

dwheeler @ ida.org

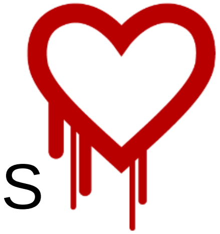
Personal: dwheeler @ dwheeler.com,
Twitter: drdavidawheeler
www.dwheeler.com

Open source software

- OSS: software licensed to users with these freedoms:
  - to *run* the program for any purpose,
  - to *study* and *modify* the program, and
  - to freely *redistribute* copies of either the original or modified program (without royalties to original author, etc.)
- Original term: “Free software” (confused with no-price)
- Other synonyms: libre sw, free-libre sw, FOSS, FLOSS
- Antonyms: proprietary software, closed software
- Widely used; OSS #1 or #2 in many markets
  - “… plays a more critical role in the DoD than has generally been recognized.” [MITRE 2003]
- OSS almost always *commercial* by law & regulation
  - Software licensed to general public & has non-government use → commercial software (in US law, per 41 USC 403)

Background

- It is *not* the case that “all OSS* is insecure” … or that “all OSS is secure”
  - Just like all other software, some OSS is (relatively) secure.. and some is not
- Heartbleed vulnerability in OpenSSL
  - Demonstrated in 2014 that some widely-used OSS needs investment for security



- Linux Foundation created Core Infrastructure Initiative (CII) in 2014
  - “to fund and support critical elements of the global information infrastructure”

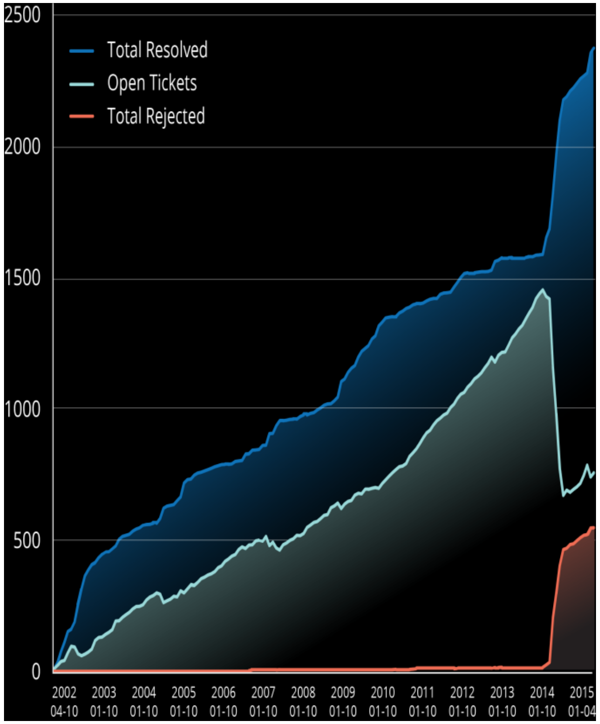
*OSS=Open source software

A little about the CII

- Multi-million dollar project
  - Supported by many, e.g., Amazon Web Services, Adobe, Bloomberg, Cisco, Dell, Facebook, Fujitsu, Google, Hitachi, HP, Huawei, IBM, Intel, Microsoft, NetApp, NEC, Qualcomm, RackSpace, salesforce.com, and VMware
- Actions
  - Collaboratively identifies & funds OSS projects in need of assistance
  - Allows developers to continue their work under community norms
  - Transitioning from point fixes to holistic solutions for open source security

CII-funded investments in key OSS projects

- OpenSSL
  - Funded key developers: improving security, enabling outside reviews, & improving responsiveness
  - Working with the Open Crypto Audit Project, has retained the NCC Group to audit OpenSSL code
- OpenSSH
- GnuPG
- Network Time Protocol (NTP) daemon
- Linux Kernel Self Protection Project
- …



OpenSSL issues

Source: https://www.coreinfrastructure.org/grants

CII-funded projects with multi-project impacts

- The fuzzing project
- OWASP Zed Attack Proxy (ZAP) as a service
- False-Positive-Free Testing with Frama-C
- Reproducible builds
- CII census (project quantitative analysis)
- Best practices badge (focus today)

Source: https://www.coreinfrastructure.org/grants

CII Best Practices Badge

- OSS tends to be more secure if it follows good security practices, undergoes peer review, etc.
  - How can we encourage good practices?
  - How can we know good practices are being followed?
- Badging project approach:
  - Identified a set of best practices for OSS projects
    - Best practices is for OSS projects (*production* side)
    - Based on existing materials & practices
  - Created web application: OSS projects self-certify
    - If OSS project meets criteria, it gets a badge (scales!)
    - Self-certification problems mitigated by automation, public display of answers (for criticism), LF can override

Badge scoring system

- To obtain a badge, all:
  - MUST and MUST NOT criteria (42/66) must be met
  - SHOULD (10/66) met, OR unmet with justification
    - Users can see those justifications & decide if that’s enough
  - SUGGESTED (14/66) considered (met or unmet)
    - People don’t like admitting they didn’t do something
  - In some cases, URL required in justification (to point to evidence; 8/66 require this)
- Currently one level (passing vs. in progress)
  - Capture what well-run projects typically already do
    - Not “they should do X, but no one does that”
  - Intend to later add higher levels with stronger requirements (gold/platinum?). ~annual updates

Criteria categories and examples (1)

1. Basics

- The software MUST be released as FLOSS*. [floss_license]
- It is SUGGESTED that any required license(s) be approved by the Open Source Initiative (OSI). [floss_license_osi]

2. Change Control

- The project MUST have a version-controlled source repository that is publicly readable and has a URL. [repo_public]
  - Details: The URL MAY be the same as the project URL. The project MAY use private (non-public) branches in specific cases while the change is not publicly released (e.g., for fixing a vulnerability before it is revealed to the public).

3. Reporting

- The project MUST publish the process for reporting vulnerabilities on the project site. [vulnerability_report_process]

*FLOSS=Free/Libre/Open Source Software

Criteria categories and examples (2)

4. Quality

- If the software requires building for use, the project MUST provide a working build system that can automatically rebuild the software from source code. [build]
- The project MUST have at least one automated test suite that is publicly released as FLOSS (this test suite may be maintained as a separate FLOSS project). [test]
- The project MUST have a general policy (formal or not) that as major new functionality is added, tests of that functionality SHOULD be added to an automated test suite. [test_policy]
- The project MUST enable one or more compiler warning flags, a "safe" language mode, or use a separate "linter" tool to look for code quality errors or common simple mistakes, if there is at least one FLOSS tool that can implement this criterion in the selected language. [warnings]

Criteria categories and examples (3)

5. Security

- At least one of the primary developers MUST know of common kinds of errors that lead to vulnerabilities in this kind of software, as well as at least one method to counter or mitigate each of them. [know_common_errors]
- The project's cryptographic software MUST use only cryptographic protocols and algorithms that are publicly published and reviewed by experts. [crypto_published]
- The project MUST use a delivery mechanism that counters MITM attacks. Using https or ssh+scp is acceptable. [delivery_mitm]
- There MUST be no unpatched vulnerabilities of medium or high severity that have been publicly known for more than 60 days. [vulnerabilities_fixed_60_days]

Criteria categories and examples (4)

6. Analysis

- At least one static code analysis tool MUST be applied to any proposed major production release of the software before its release, if there is at least one FLOSS tool that implements this criterion in the selected language… [static_analysis]
- It is SUGGESTED that the {static code analysis} tool include rules or approaches to look for common vulnerabilities in the analyzed language or environment. [static_analysis_common_vulnerabilities]
- It is SUGGESTED that at least one dynamic analysis tool be applied to any proposed major production release of the software before its release. [dynamic_analysis]

Current state

- General availability announced May 2016
- As of 2016-09-14: 280 project entries
  - 35 are passing (100%), 63 are 90%+ (incl. 100%)
- Examples of current badge holders:
  - BadgeApp (itself!)
  - Node.js
  - Linux kernel
  - curl
  - GitLab
  - OpenSSL (pre-Heartbleed missed 1/3 criteria)
  - Zephryr project

Source: https://bestpractices.coreinfrastructure.org/projects

Sample impacts of CII badge process

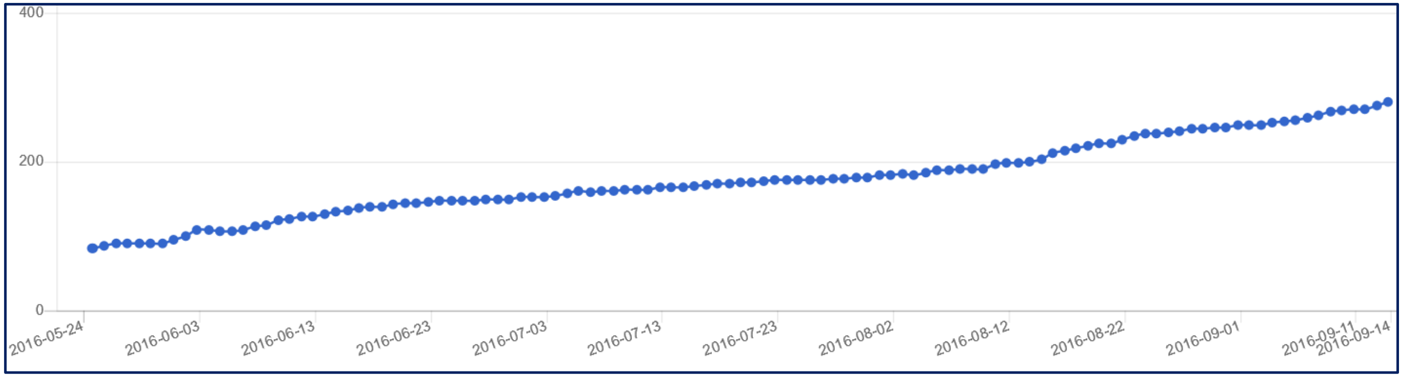
- OWASP ZAP (web app scanner)
  - Simon Bennetts: “[it] helped us improve ZAP quality… [it] helped us focus on [areas] that needed most improvement.”
  - Change: Significantly improved automated testing
- CommonMark (Markdown in PHP) changes:
  - TLS for the website (& links from repository to it)
  - Publishing the process for reporting vulnerabilities

  } common issues
- OPNFV (open network functions virtualization)
  - Change: Replaced no-longer-secure crypto algorithms
- JSON for Modern C++
  - “I really appreciate some formalized quality assurance which even hobby projects can follow.”
  - Change: Added explicit mention how to privately report errors
  - Change: Added a static analysis check to continuous integration script

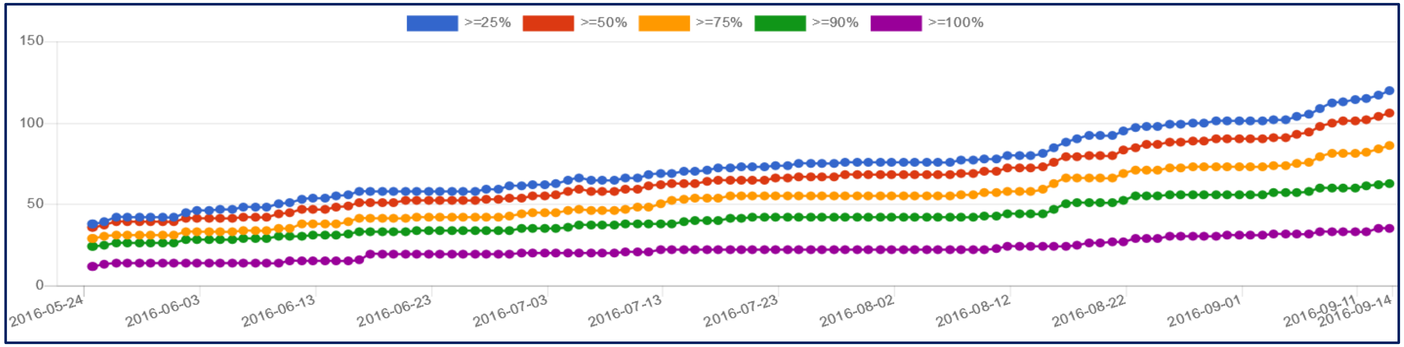
Source: https://github.com/linuxfoundation/cii-best-practices-badge/wiki/Impacts

CII badges are getting adopted!

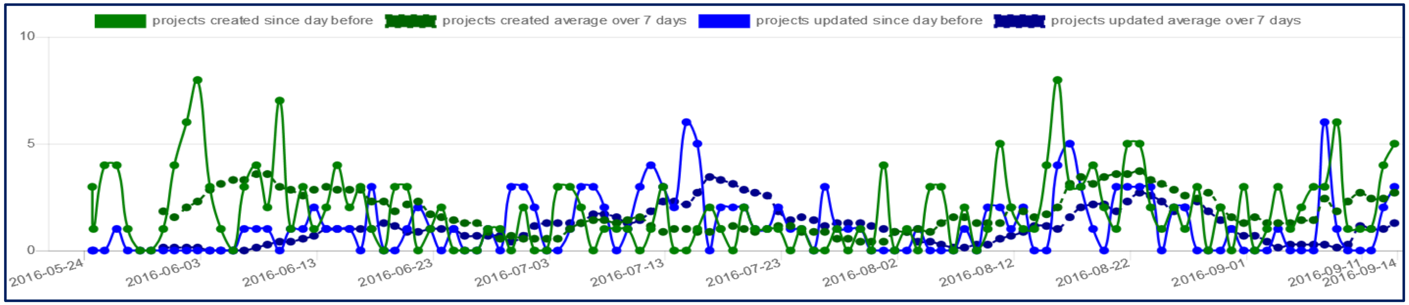
All projects



Projects with non-trivial progress



Daily activity



Source: https://bestpractices.coreinfrastructure.org/project_stats

BadgeApp security

- File “security.md” describes how we secure the web app
- Report vulnerabilities to designated people
- Requirements – simple, most data public
  - Passwords stored in database only as iterated salted hashes
- Design: Showed that we applied design principles
  - Simple, filter inputs
- Implementation
  - Checked that it counters all of OWASP top 10
  - Applied “Ruby on Rails Security Guide”
  - Hardened (e.g., hardening HTTP headers)
- Verification
  - Source code quality analyzer (rubocop, rails_best_practices), [static] source code weakness analyzer (brakeman), web application scanner (OWASP ZAP), 98% test coverage, OSS enables multi-person review
- Supply chain (reuse)
  - Consider before use, bundle-audit (check known vulns), license_finder
  - Strive to minimize/simplify transitive dependencies & size
- People

Future criteria, gold/platinum levels

- Probable future “passing” criteria include:
  - It is SUGGESTED that hardening mechanisms be used so software defects are less likely to result in security vulnerabilities. [hardening]
  - It is SUGGESTED that the project have a reproducible build. [build_reproducible]
- Some potential gold/platinum criteria (doc/other.md):
  - Active development community
  - Bus factor >= 2
  - Dependencies (including embedded dependencies) are periodically checked for known vulnerabilities, & updated or verified as unexploitable
  - All changes reviewed by someone else before release
  - Automated test suite has 100% branch coverage of source code
  - Move SHOULD/SUGGESTED to MUST

Involved in OSS?

- If you lead an OSS project, what you do matters!
  - People depend on the software you create
  - The practices you apply affect the result
  - Secure or quality software is not an accident
- If you’re considering using an OSS project
  - Check on the project – should you use it?
- Badge criteria help

Key URLs

- CII
  - https://www.coreinfrastructure.org
- CII best practices badge (get a badge):
  - https://bestpractices.coreinfrastructure.org/
- CII best practices badge project:
  - https://github.com/linuxfoundation/cii-best-practices-badge

Backup

Mozilla Open Source Support (MOSS) relation

- Mozilla Open Source Support (MOSS) added Secure Open Source (SOS) track
    - Announced June 9, 2016
    - “supports security audits for open source software projects, and remedial work to rectify the problems found”
    - “support model is different from & complementary to CII. [CII focuses on] deeper-dive investments into core OS security infrastructure, while [SOS targets] OSS projects with lower-hanging fruit security needs.”
- CII complements other efforts like MOSS

Sources: https://wiki.mozilla.org/MOSS/Secure_Open_Source
https://blog.mozilla.org/blog/2016/06/09/help-make-open-source-secure/

Badge criteria must be…

- Relevant
- Attainable by typical OSS projects
- Clear
- Include security-related criteria
- Consensus of developers & users
  - Criteria & web app developed as OSS project
  - Built on existing work, e.g., Karl Fogel’s *Producing Open Source Software*
- Not hypocritical
  - Our web app must get its own badge!

Worked with several projects, including the Linux kernel & curl, to perform alpha test of criteria

Badge criteria must NOT be…

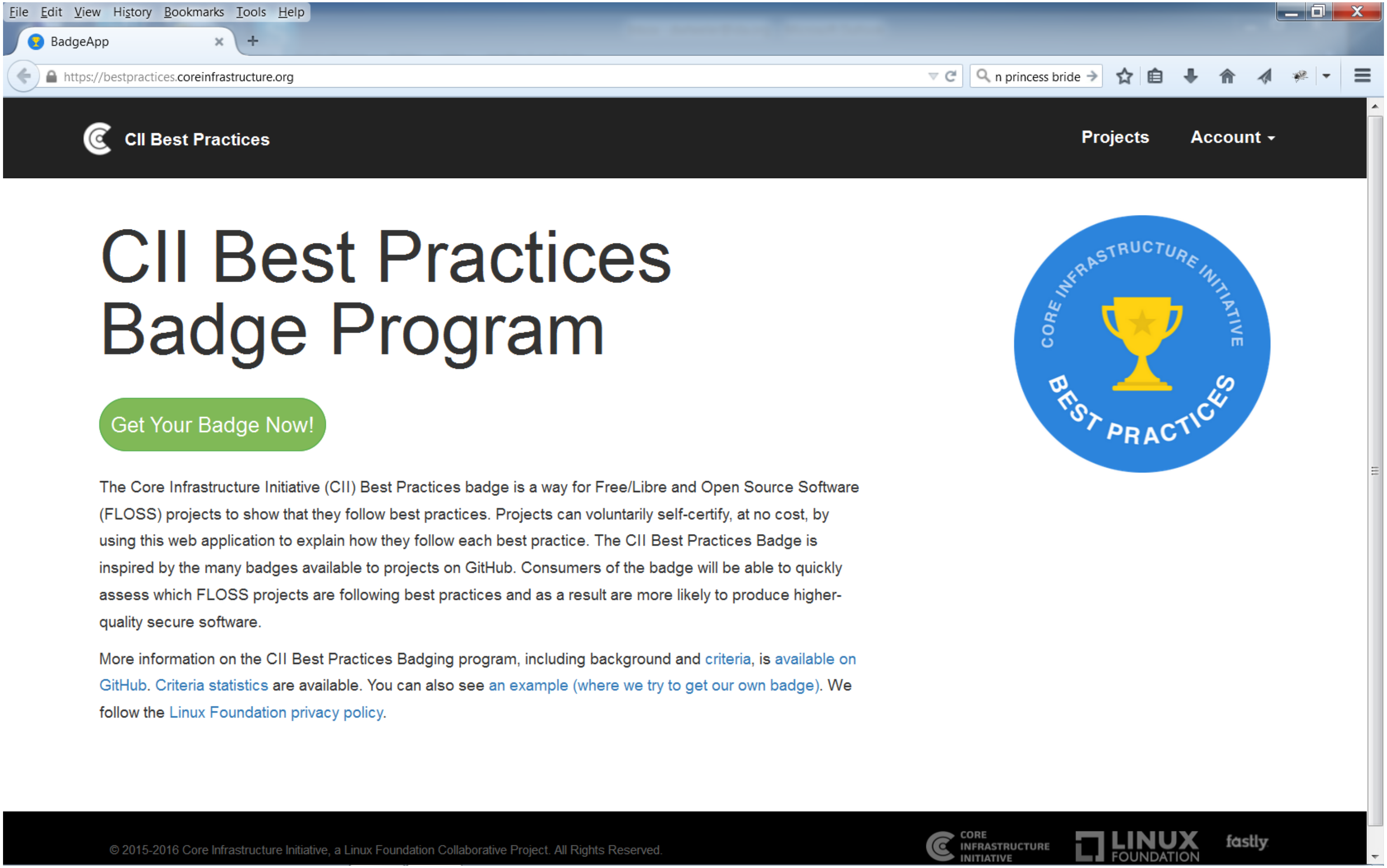
- Will NOT require any specific products or services (especially proprietary ones)
  - We intentionally don’t require git or GitHub
  - That said, will automate many things if project *does* use GitHub
- Will NOT require or forbid any particular programming language

Describing criteria

- Criteria have different levels of importance
  - MUST (NOT) – required (42/66)
  - SHOULD (NOT) – sometimes valid to not do (10/66)
  - SUGGESTED – common valid reasons, but at least consider it (14/66)
- Criteria may have “details” (39/66)
  - Give clarifications/examples, e.g., “MAY…”
- Each criterion is named (lowercase underscore)
- For each criterion, users answer:
  - Status: Met, Unmet, Unknown (?), N/A*
  - Justification: Markdown text. Usually optional

* N/A is only allowed for 21/66 criteria

BadgeApp: Home page



File Edit View History Bookmarks Tools Help
BadgeApp
https://bestpractices.coreinfrastructure.org
n princess bride
CII Best Practices
Projects
Account
CORE INFRASTRUCTURE INITIATIVE
BEST PRACTICES

CII Best Practices Badge Program

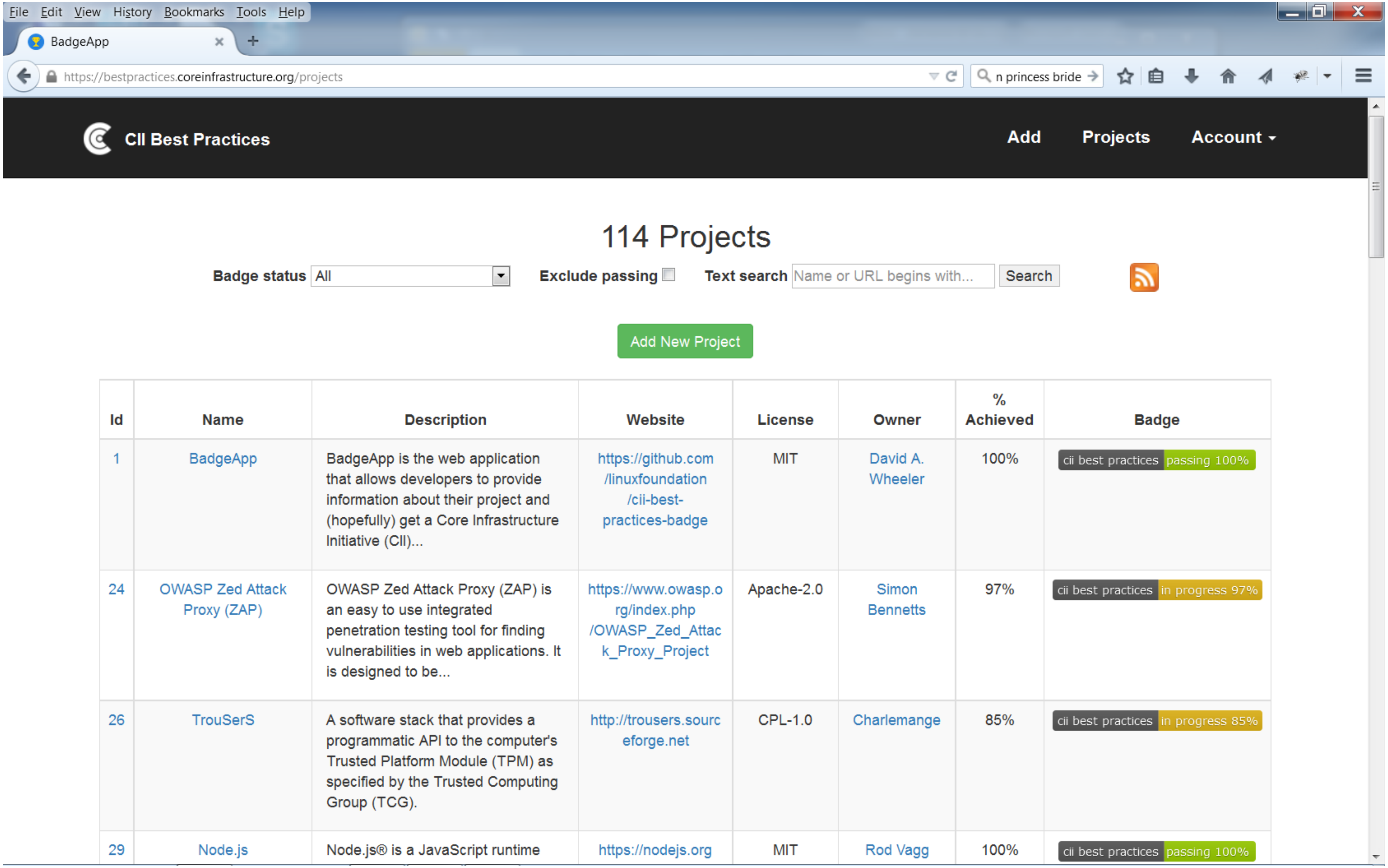
Get Your Badge Now!

The Core Infrastructure Initiative (CII) Best Practices badge is a way for Free/Libre and Open Source Software (FLOSS) projects to show that they follow best practices. Projects can voluntarily self-certify, at no cost, by using this web application to explain how they follow each best practice. The CII Best Practices Badge is inspired by the many badges available to projects on GitHub. Consumers of the badge will be able to quickly assess which FLOSS projects are following best practices and as a result are more likely to produce higher-quality secure software.

More information on the CII Best Practices Badging program, including background and criteria, is available on GitHub. Criteria statistics are available. You can also see an example (where we try to get our own badge). We follow the Linux Foundation privacy policy.

© 2015-2016 Core Infrastructure Initiative, a Linux Foundation Collaborative Project. All Rights Reserved.

BadgeApp: List of projects

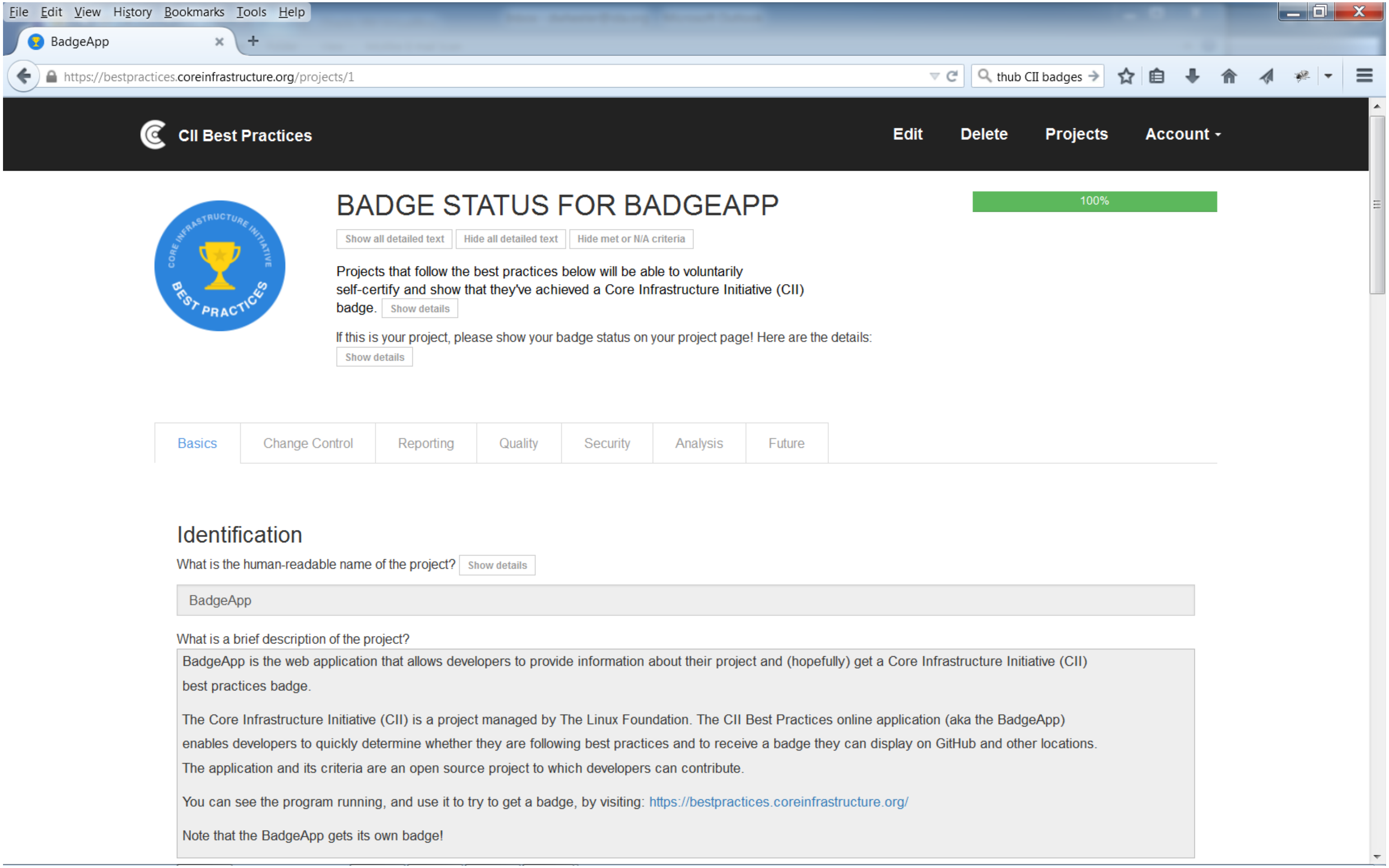

File Edit View History Bookmarks Tools Help
BadgeApp
https://bestpractices.coreinfrastructure.org/projects
n princess bride
CII Best Practices
Add
Projects
Account
Add New Project
Search

114 Projects

Badge status: All | Exclude passing ☐ | Text search: Name or URL begins with...

Id	Name	Description	Website	License	Owner	% Achieved	Badge
1	BadgeApp	BadgeApp is the web application that allows developers to provide information about their project and (hopefully) get a Core Infrastructure Initiative (CII)...	https://github.com/linuxfoundation/cii-best-practices-badge	MIT	David A. Wheeler	100%	cii best practices passing 100%
24	OWASP Zed Attack Proxy (ZAP)	OWASP Zed Attack Proxy (ZAP) is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications. It is designed to be...	https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project	Apache-2.0	Simon Bennetts	97%	cii best practices in progress 97%
26	TrouSerS	A software stack that provides a programmatic API to the computer's Trusted Platform Module (TPM) as specified by the Trusted Computing Group (TCG).	http://trousers.sourceforge.net	CPL-1.0	Charlemange	85%	cii best practices in progress 85%
29	Node.js	Node.js® is a JavaScript runtime	https://nodejs.org	MIT	Rod Vagg	100%	cii best practices passing 100%

BadgeApp: Itself as a sample project



BADGE STATUS FOR BADGEAPP

Projects that follow the best practices below will be able to voluntarily self-certify and show that they've achieved a Core Infrastructure Initiative (CII) badge.

If this is your project, please show your badge status on your project page! Here are the details:

Identification

What is the human-readable name of the project?

BadgeApp

What is a brief description of the project?

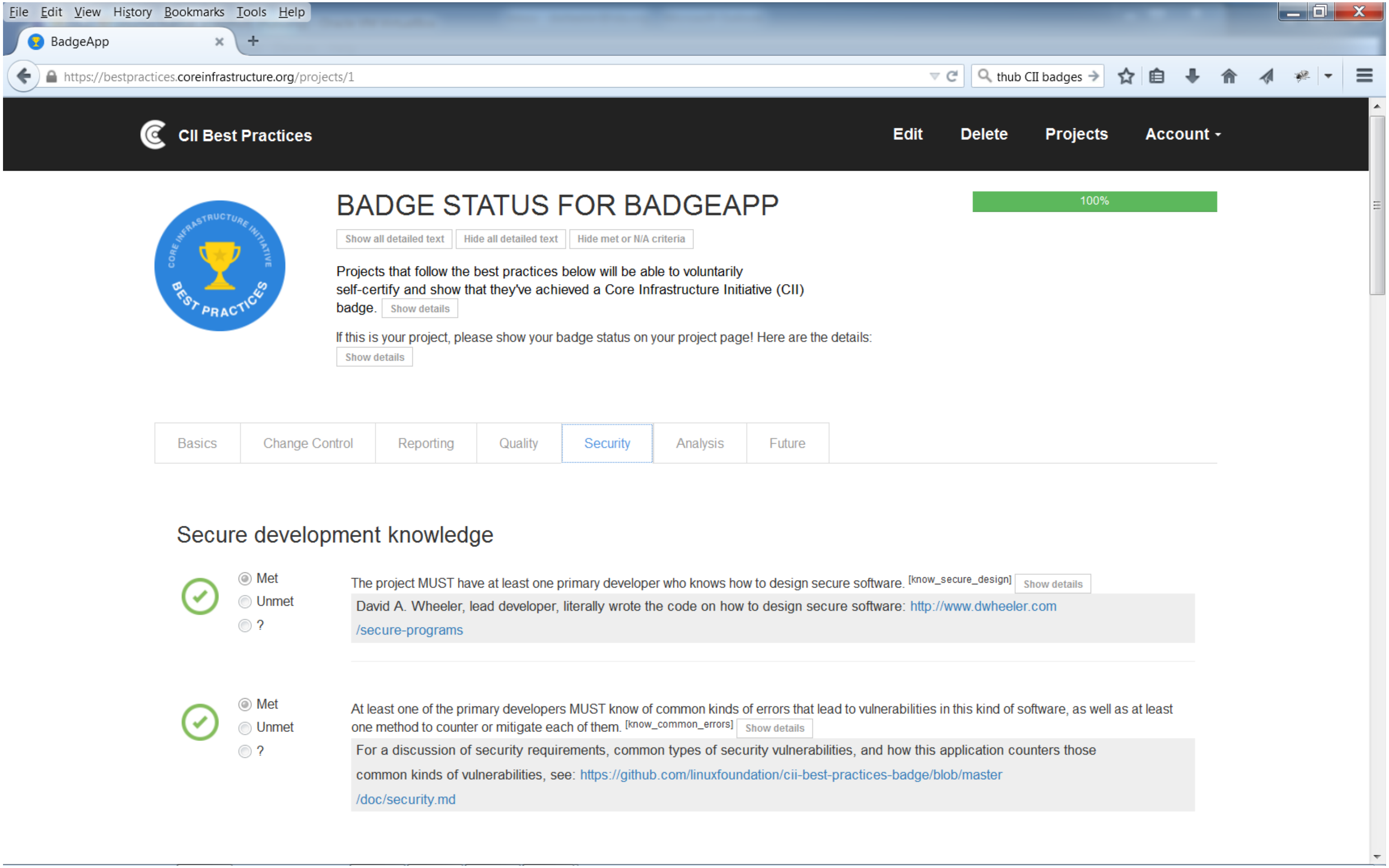
BadgeApp is the web application that allows developers to provide information about their project and (hopefully) get a Core Infrastructure Initiative (CII) best practices badge.

The Core Infrastructure Initiative (CII) is a project managed by The Linux Foundation. The CII Best Practices online application (aka the BadgeApp) enables developers to quickly determine whether they are following best practices and to receive a badge they can display on GitHub and other locations. The application and its criteria are an open source project to which developers can contribute.

You can see the program running, and use it to try to get a badge, by visiting: https://bestpractices.coreinfrastructure.org/

Note that the BadgeApp gets its own badge!

BadgeApp: Sample project (security tab)



BADGE STATUS FOR BADGEAPP

Projects that follow the best practices below will be able to voluntarily self-certify and show that they've achieved a Core Infrastructure Initiative (CII) badge. Show details

If this is your project, please show your badge status on your project page! Here are the details: Show details

Secure development knowledge

✓ ● Met ○ Unmet ○ ?

The project MUST have at least one primary developer who knows how to design secure software. [know_secure_design] Show details

David A. Wheeler, lead developer, literally wrote the code on how to design secure software: http://www.dwheeler.com/secure-programs

✓ ● Met ○ Unmet ○ ?

At least one of the primary developers MUST know of common kinds of errors that lead to vulnerabilities in this kind of software, as well as at least one method to counter or mitigate each of them. [know_common_errors] Show details

For a discussion of security requirements, common types of security vulnerabilities, and how this application counters those common kinds of vulnerabilities, see: https://github.com/linuxfoundation/cii-best-practices-badge/blob/master/doc/security.md

EU-FOSSA project interactions with CII Badge

- EU-FOSSA = EU-Free and Open Source Software Auditing
  - 1M Euro project initiated by 2 Members of European Parliament
  - Executed by European Commission (the European Union's executive body)
  - Goal: invest into commonly used OSS which might need support in the security domain
- Intends to define a complete process to properly perform code reviews within the European Institutions
  - To execute one sample code review during Q3-Q4/2016
  - Sample results will determine if activity could become a continuous action of the European Institutions in the future
- FOSSA project exchanging experiences with CII
- FOSSA looking closely at the CII Badge criteria
  - During definition of metrics to analyze sustainability and security

See: https://joinup.ec.europa.eu/community/eu-fossa/description and https://fosdem.org/2016/schedule/event/fossa/

A few notes on the BadgeApp

- “BadgeApp” is simple web application that implements the criteria (fill in form)
  - OSS (MIT license)
    - All libraries OSS & legal to add (checked with license_finder)
  - Simple Ruby on Rails app
  - Criteria info (text, category, etc.) in YAML
- Overall approach: Proactively counter mistakes
  - Mistakes happen; we use a variety of tools, automated test suite, processes to counter them
- Please contribute!
  - See its CONTRIBUTING.md for more